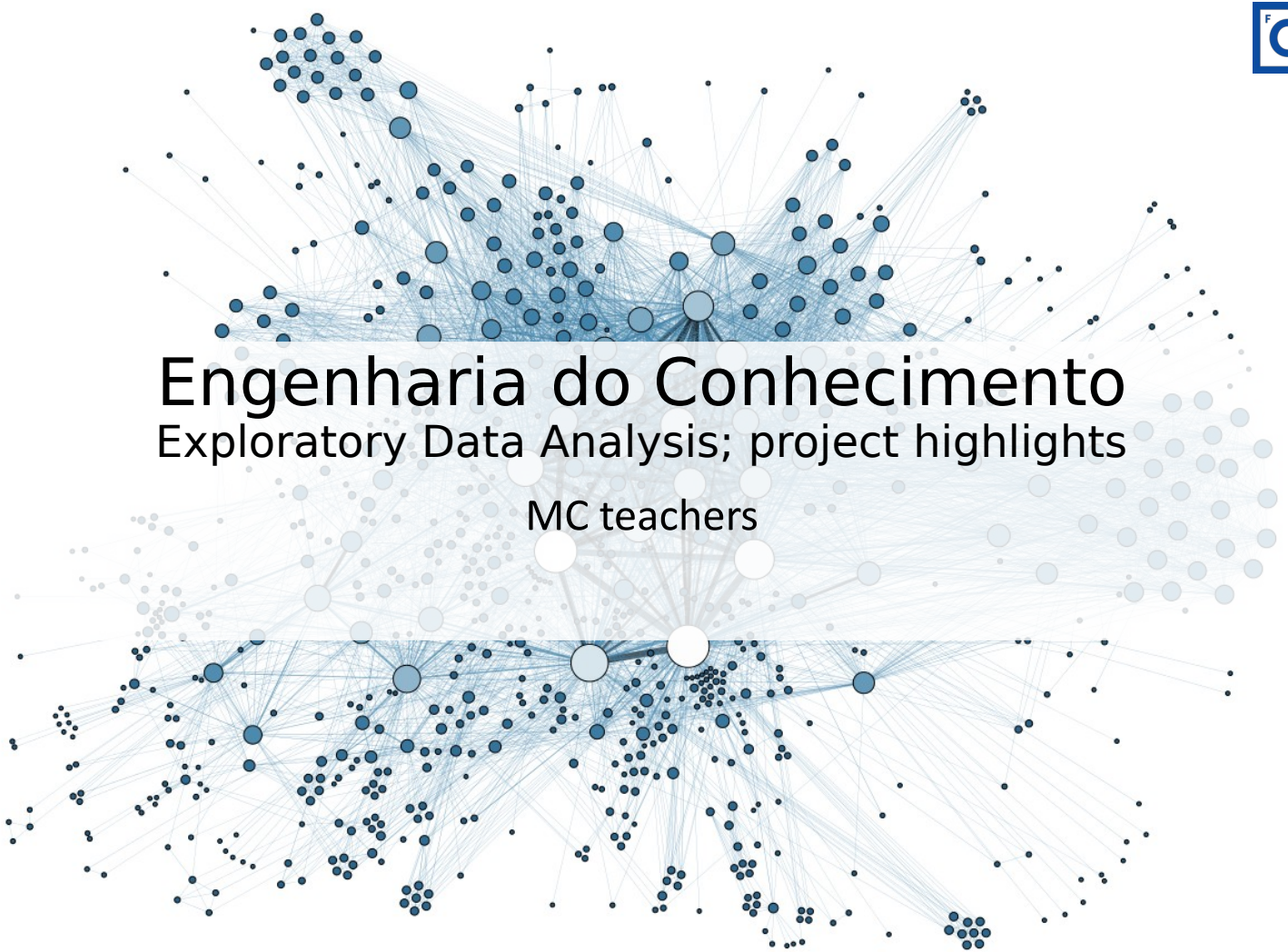

Engenharia do Conhecimento

Exploratory Data Analysis; project highlights

MC teachers

know your data

Exploratory Data Analysis (EDA)

a data profiler task

helps in

- identifying variable types
- identifying missing values and 1D outlier data points
- understanding the relationships between the various attributes
- and between the attributes and the target
- recognizing the important attributes
- visualizing the variable data distributions
- choosing a machine learning algorithm for the problem

data profiler

a tool that

- automatically analyses the data

  and

- produces statistics and correlations

- an example: pandas profiling, now **ydata profiling**

In [1]:
```
from sklearn.datasets import load_diabetes
import pandas as pd

diabetes = load_diabetes()
# Convert to a pandas dataframe
df = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
# Add the target variable to the dataframe
df['target'] = diabetes.target
df
```

Out[1]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0
...	...	...	...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.031193	0.007207	178.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.018114	0.044485	104.0
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011080	-0.046883	0.015491	132.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.044529	-0.025930	220.0
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039493	-0.004222	0.003064	57.0

442 rows × 11 columns

In [2]:
```
from ydata_profiling import ProfileReport

profile = ProfileReport(df)
profile.to_file("diabetes_profile.html")
```

Summarize dataset: 100% 120/120 [00:09<00:00, 9.36it/s, Completed]

Generate report structure: 100% 1/1 [00:03<00:00, 3.47s/it]

Render HTML: 100% 1/1 [00:02<00:00, 2.35s/it]

Export report to file: 100% 1/1 [00:00<00:00, 26.62it/s]

Overview

Overview | Alerts 7 | Reproduction

Dataset statistics

Number of variables	11
Number of observations	442
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	38.1 KiB
Average record size in memory	88.3 B

Variable types

Numeric	10
Categorical	1

Variables

Select Columns

things to notice from profiler analysis

“diabetes” is a well curated and prepared dataset

- no missing values
- no duplicate rows
- normalised features (except ‘target’) – what kind of normalisation?...

what the profiler provides per variable

- statistics
- histogram
- correlations (2x2) – including target variable in dataframe is helpful

know your tools

some ML issues... of scikit-learn

train

test

some ML issues... of scikit-learn

An example to remember:

- decision trees don’t have any theoretical limitation on mixing types of attributes
- **however**, scikit-learn internally converts all attribute values to dtype=np.float32

  ⇒ categorical attributes must first be encoded to numeric

- classes **OrdinalEncoder** and **OneHotEncoder** can be used for attributes

train

test

some ML issues... of scikit-learn

An example to remember:

- decision trees don't have any theoretical limitation on mixing types of attributes
- **however**, scikit-learn internally converts all attribute values to dtype=np.float32

  ⇒ categorical attributes must first be encoded to numeric

- classes **OrdinalEncoder** and **OneHotEncoder** can be used for attributes

notice: you must encode train and test data with the same encoder!

```
enc1.fit_transform(X_train)   train

enc1.transform(X_test)        test
```

& some ML goodies... of scikit-learn

```
from sklearn.pipeline import Pipeline
```

& some ML goodies... of scikit-learn

- Pipeline

```
from sklearn.pipeline import Pipeline
```

- Allows the composition of several operations to be defined and called at once
- Particularly useful for data preparation

know your project

24/25 project’s dataset

Read data/Problem description

Read data/Problem description

- The data is composed of 21 attribute values
- Section 2 has the description of each attribute values:
    - Boolean values
    - Continuous values
    - Missing data (97, 98, 99)
- 4 objectives of the two types
    - Classification
    - Regression

Read project description (in moodle)

Read project description (in moodle)

- Last cell in handed-in notebook must have tests of models with the **test set** (mock-ups are provide

```
In [8]: X_test = pd.read_csv("proj-test-data.csv", sep=",")
        y_text = pd.read_csv("proj-test-class.csv", sep=",")
        # apply models to the test set
        # and show results!
```

use these names!
(of the supplied mock-up files)

Project development

- Everything you did in the project will be evaluated
- Steps followed
- Data preparation used
- Models used
- Metrics used
- Make sure you describe the rationale of your options in the report!

enjoy!

Next class

- neural networks – take 1